Score Matching & Diffusion Models

Chitwan Saharia
Ideogram AI

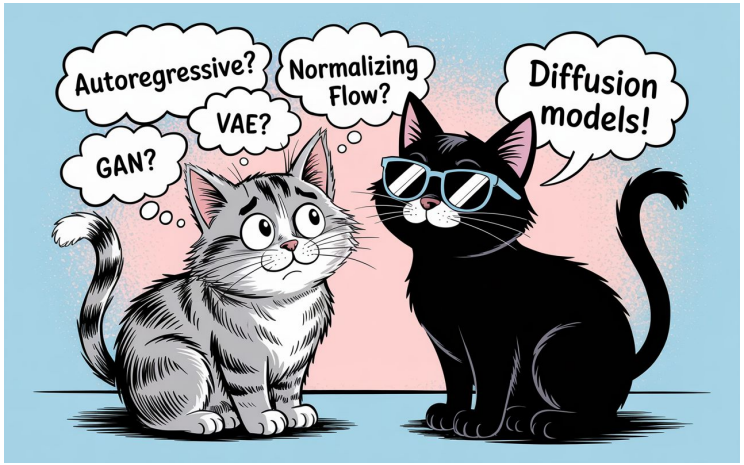
Objective

Goal

Given a large corpus of images, train a generative model such that we can sample new images from this model.

How can we do this?

- **Autoregressive model**
- GAN
- Normalizing Flows
- VAE



Autoregressive Model

$$p(x) = \prod_{i=1}^{n} p(x_{\pi_i} | x_{\pi_1}, ..., x_{\pi_{i-1}}, \theta)$$

Pros:
- Tractable exact log likelihood maximization

Cons:
- This factorization requires a causal model and a tractable distribution for each factor.
- Generates “tokens” one at a time.

Autoregressive Model: Method 1

1680 x 1122 x 3 matrix
where each value is in {0, 1, 2, … 255}

1680 x 1122 x 3 ~ **5.65M tokens** with vocabulary size of 256

Generative Pretraining from Pixels

Autoregressive Model: Method 2 (More Common)

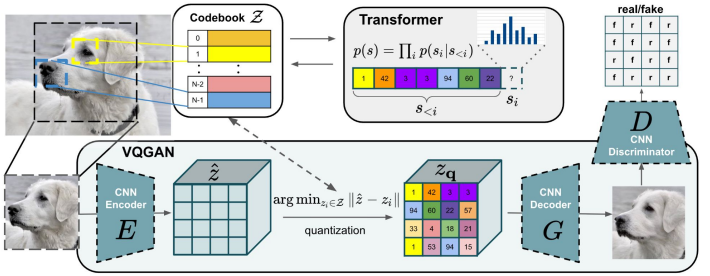


Figure 2. Our approach uses a convolutional *VQGAN* to learn a codebook of context-rich visual parts, whose composition is subsequently modeled with an autoregressive transformer architecture. A discrete codebook provides the interface between these architectures and a patch-based discriminator enables strong compression while retaining high perceptual quality. This method introduces the efficiency of convolutional approaches to transformer based high resolution image synthesis.

[Taming Transformers for High-Resolution Image Synthesis]

Encoder: Take an image, map each patch (e.g. 16x16x3) to an embedding in a 32K embedding table (codebook)

Decoder: Take the mapped embedding lookups, and reconstruct the original image as faithfully as possible.

Autoregressive Model: Method 2 (More Common)

1680 x 1122 x 3 matrix

(1680 / 16) x (1122 / 16) matrix

7350 tokens (32K vocab size)

Encoder: Take an image, map each patch (e.g. 16x16x3) to an embedding in a 32K embedding table (codebook)

Decoder: Take the mapped embedding lookups, and reconstruct the original image as faithfully as possible.

Scaling Autoregressive Models for Content-Rich Text-to-Image Generation

Autoregressive Model: Method 2 (More Common)

This approach has been shown to work successfully.

Zero-Shot Text-to-Image Generation - Original DALL-E paper

Scaling Autoregressive Models for Content-Rich Text-to-Image Generation

Chameleon: Mixed-Modal Early-Fusion Foundation Models

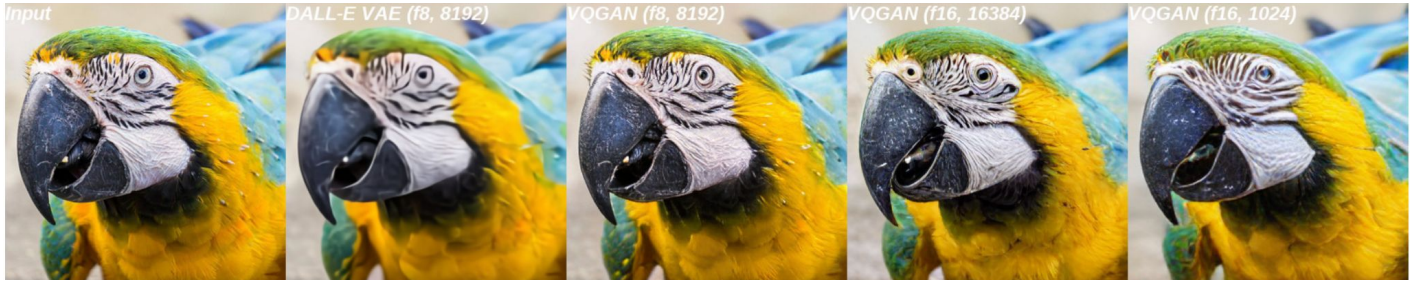
Autoregressive Model: Method 2 (More Common)

Main Limitation

The mapping of images to 32K discrete tokens is lossy.

16 x 16 x 3 pixel patch → 768 bits being mapped to ~15 bits

Hence, even a perfect autoregressive model is bound by the lossiness of the autoencoder.



Score Matching and Diffusion Models

Diffusion Models Beat GANs on Image Synthesis

WAVEGRAD: ESTIMATING GRADIENTS FOR WAVEFORM GENERATION

Palette: Image-to-Image Diffusion Models

Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

Image Super-Resolution via Iterative Refinement

Text Generation with Diffusion Language Models: A Pre-training Approach with Continuous Paragraph Denoise

Cascaded Diffusion Models
for High Fidelity Image Generation

All-Atom Protein Generation with Latent Diffusion

IMAGEN VIDEO: HIGH DEFINITION VIDEO GENERATION WITH DIFFUSION MODELS

Score Based Generative Models

What is a score?

Say we have a random variable X with a probability distribution $p(x)$, then the score function of $p(x)$ is defined as

$$\nabla_x \log p(x)$$

Notes:

- **X** must be a **continuous** random variable, for score to be defined.
- We can treat an image as an instance of a **continuous** random variable, even though technically, images are represented as discrete objects.

Score Based Generative Models

What is a score?

Say we have a random variable X with a probability distribution $p(x)$, then the score function of $p(x)$ is defined as

$$\nabla_x \log p(x)$$

Core Idea:

- Given a set of data points from the distribution p(x), train a neural network $s_\theta(x)$ that can map to this score function.
- Once we have this score function estimator $s_\theta(x)$, we can use this to generate new samples from the model using *Langevin dynamics.*

Quick Detour: Langevin Dynamics

Given a probability distribution ***p(x)***, langevin dynamics provides a Markov Chain Monte Carlo (MCMC) to obtain random samples from this distribution.

Say x_0 ~ *Q(x)* where *Q(x)* can be any arbitrary distribution which is easy to sample from (e.g. standard Normal Gaussian distribution), we can sample from *p(x)* using the following iterative process

$$x_{t+1} = x_t + \epsilon \nabla \log_{x_t} p(x_t) + \sqrt{2}\epsilon z_i$$

where z_t ~ N(0, I). As long as the step size a.k.a \epsilon → 0, and the number of iterations → \infinity, the resulting sample x_t converges to a sample from ***p(x)***.

Score Based Generative Models

How do we learn $s_\theta(x)$?

Essentially, this is the objective function we care about minimizing.

$$\mathbb{E}_{p(\mathbf{x})}[\|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2] = \int p(\mathbf{x}) \|\nabla_{\mathbf{x}} \log p(\mathbf{x}) - \mathbf{s}_\theta(\mathbf{x})\|_2^2 \mathrm{d}\mathbf{x}.$$

Just like typical maximum likelihood objectives, the expectation can be estimated by Monte Carlo estimation through the available data samples.

There are two main problems here:

- We do not have access to the ground truth score values.
- The above objective function would not let us train the score function on any data point outside the manifold. i.e. if *p(x) ~ 0.*

Score Based Generative Models

Say we start $x_0 \sim N(0, I)$, which clearly will have $p(x) \sim 0$, the score function $s_\theta(x)$ will essentially be random, and not necessarily pointing towards the data distribution.

$$x_{t+1} = x_t + \epsilon \nabla \log_x p(x) + \sqrt{2}\epsilon z_i$$

Score Based Generative Models

Say we start $x_0 \sim N(0, I)$, which clearly will have $p(x) \sim 0$, the score function $s_\theta(x)$ will essentially be random, and not necessarily pointing towards the data distribution.

$$x_{t+1} = x_t + \epsilon \nabla \log_x p(x) + \sqrt{2}\epsilon z_i$$

Solution:

Let's define N distributions $p_{\sigma_0}(\bar{x}), p_{\sigma_1}(\bar{x}), p_{\sigma_2}(\bar{x}), ...p_{\sigma_N}(\bar{x})$ where

$$p_{\sigma_i}(\bar{x}) = \int p(x)\mathcal{N}(\bar{x}; x, \sigma_i^2 I)$$

and $\sigma_0 \sim 0 < \sigma_1 < \sigma_2 < ..... < \sigma_N$

Score Based Generative Models

Combining all the different noise levels, we end up with this new objective function

$$\sum_{i=0}^{N} \lambda(i) \int p_{\sigma_i}(\bar{x})[||\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}) - s_\theta(\bar{x}, \sigma_i)||_2^2]$$

- Now, the score function **is also conditioned on the noise level** of the current distribution.
- \lambda(i) is the loss weight for the distribution i.

There is still one problem: we do not have access to the ground truth scores.

Generative Modeling by Estimating Gradients of the Data Distribution

Score Based Generative Models

A beautiful result from A connection between score matching and denoising autoencoders shows that the equivalence between these two objective functions:

$$\sum_{i=0}^{N} \lambda(i) \int p_{\sigma_i}(\bar{x}) [|| \boxed{\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x})} - s_\theta(\bar{x}, \sigma_i)||_2^2]$$

This is intractable

$$\longrightarrow$$

$$\sum_{i=0}^{N} \lambda(i) \int p_{\sigma_i}(\bar{x}) [|| \boxed{\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x)} - s_\theta(\bar{x}, \sigma_i)||_2^2]$$

This is tractable

Score Based Generative Models

$$p_{\sigma_i}(\bar{x}|x) = Ce^{-\frac{(\bar{x}-x)^2}{2\sigma_i^2}}$$

$$\log p_{\sigma_i}(\bar{x}|x) = \log C - \frac{(\bar{x}-x)^2}{2\sigma_i^2}$$

$$\boxed{\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x) = \frac{(x-\bar{x})}{\sigma_i^2}}$$

Score Based Generative Models

$$p_{\sigma_i}(\bar{x}|x) = Ce^{-\frac{(\bar{x}-x)^2}{2\sigma_i^2}}$$

$$\log p_{\sigma_i}(\bar{x}|x) = \log C - \frac{(\bar{x}-x)^2}{2\sigma_i^2}$$

$$\boxed{\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x) = \frac{(x-\bar{x})}{\sigma_i^2}}$$

$$\bar{x} = x + \epsilon\sigma_i \quad \epsilon \sim \mathcal{N}(0, I)$$

$$\boxed{\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x) = \frac{-\epsilon}{\sigma_i}}$$

Score Based Generative Models

$$\sum_{i=0}^{N} \lambda(i)\, \mathbb{E}_{\bar{x}\sim p_{\sigma_i}(\bar{x})}[||\nabla_{\bar{x}} \log p_{\sigma_i}(\bar{x}|x) - s_\theta(\bar{x}, \sigma_i)||_2^2]$$

$$\sum_{i=0}^{N} \lambda(i)\, \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),\, x\sim p(x)}[|| - \frac{\epsilon}{\sigma_i} - s_\theta(x + \epsilon\sigma_i, \sigma_i)||_2^2]$$

Setting \lambda(i) to \sigma_i^2 and reparameterizing score function to absorb -\sigma_i

$$\sum_{i=0}^{N} \mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),\, x\sim p(x)}[||\epsilon - s'_\theta(x + \epsilon\sigma_i, \sigma_i)||_2^2]$$

Score Based Generative Models

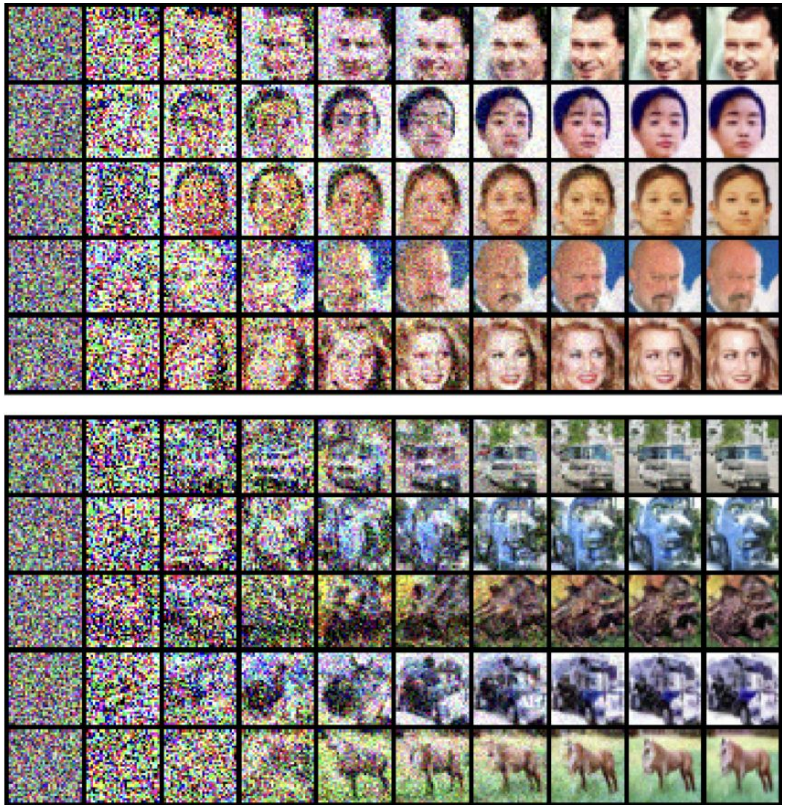
The training algorithm would look like this:

1. Given a training data point x
2. Sample one of the noise levels σ_i
3. Sample $\epsilon \sim \mathcal{N}(0, I)$
4. Minimize $||\epsilon - s'_\theta(x + \epsilon\sigma_i, \sigma_i)||_2^2$

Generative Modeling by Estimating Gradients of the Data Distribution

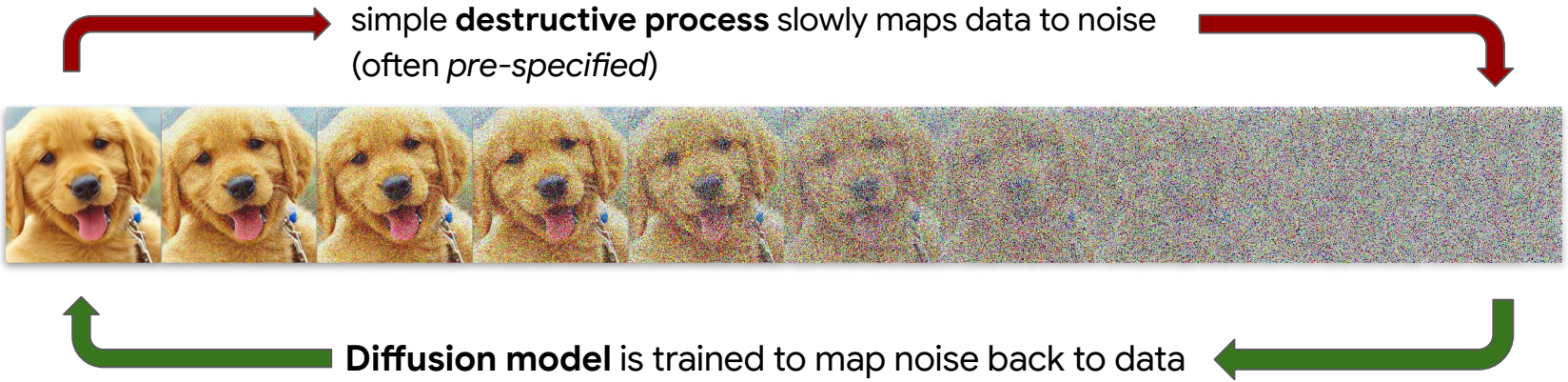
Score Based Generative Models

Visualization of intermediate samples following Langevin dynamics from a trained score model.



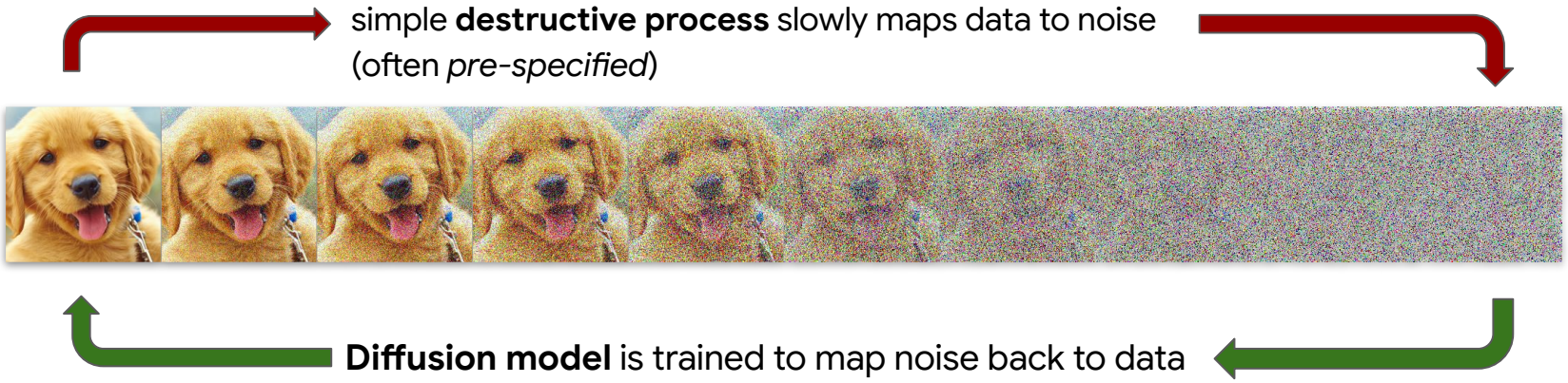
[Generative Modeling by Estimating Gradients of the Data Distribution]

Diffusion Models



[Denoising Diffusion Probabilistic Models]
[Deep Unsupervised Learning using Nonequilibrium Thermodynamics]

Diffusion Models



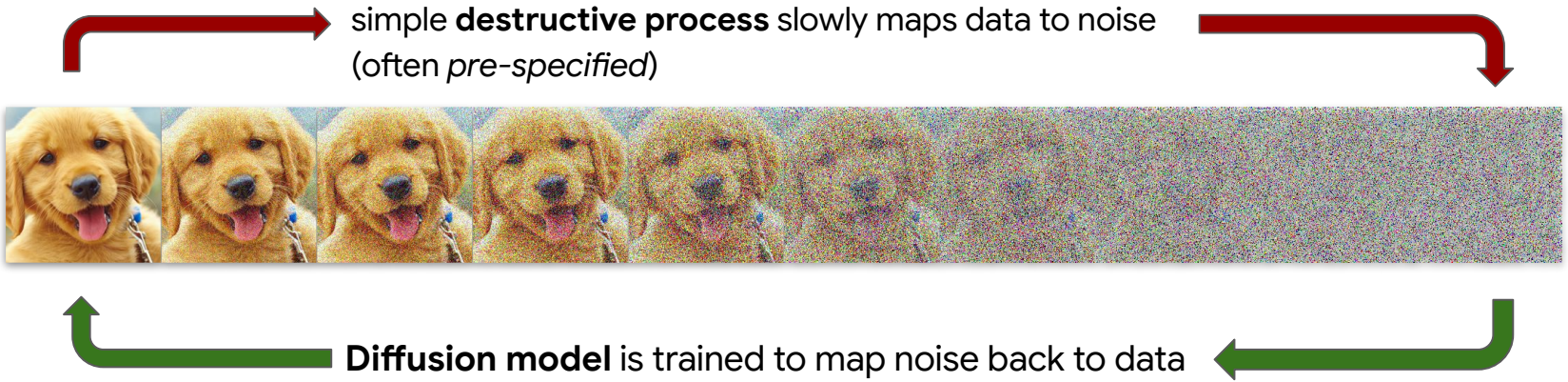
Let's define a markov chain graphical model x_0 → x_1 → x_2 → … → x_T where the following holds:

$$q(x_0) = p_{data}(x_0)$$

$$q(x_T) \approx \mathcal{N}(0, I)$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

Diffusion Models



This functional form allows us to analytically compute the distribution of x_t from x_0 (we don't have to actually run the iterative markov chain to get to x_t)

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1-\bar{\alpha}_t)I)$$

$$\bar{\alpha}_t = \prod_{i=0}^{t}(1-\beta_t)$$

Diffusion Models

Forward Process

$$q(x_0) = p_{data}(x_0)$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$q(x_T) \approx \mathcal{N}(0, I)$$

Diffusion Models

Forward Process

$$q(x_0) = p_{data}(x_0)$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$q(x_T) \approx \mathcal{N}(0, I)$$

Reverse Process

$$p_\theta(x_T) \approx \mathcal{N}(0, I)$$

$$p_\theta(x_t|x_{t+1}) = reverse(q(x_{t+1}|x_t))$$

$$p_\theta(x_0) \approx p_{data}(x_0)$$

Diffusion Models

Forward Process

$$q(x_0) = p_{data}(x_0)$$

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$$

$$q(x_T) \approx \mathcal{N}(0, I)$$

Reverse Process

$$p_\theta(x_T) \approx \mathcal{N}(0, I)$$

$$p_\theta(x_t|x_{t+1}) = reverse(q(x_{t+1}|x_t))$$

$$p_\theta(x_0) \approx p_{data}(x_0)$$

Diffusion Models

$$p_\theta(x_t|x_{t+1}) = reverse(q(x_{t+1}|x_t))$$

If we set β_t to be significantly smaller compared to the variance of x_t.

The smaller the β_t, the closer the functional form of the $\boldsymbol{p_\theta}$ is to $\boldsymbol{q}$. Given, $\boldsymbol{q}$ is a unimodal gaussian distribution, we can parameterize $\boldsymbol{p_\theta}$ as

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(x_t; \boxed{\mu_\theta(x_t, t), \sigma_\theta(x_t, t)})$$

We need to learn these two time / noise scale conditional models. For simplicity, we generally keep the variance fixed, and only learn the mean.

Diffusion Models

Training

Training amounts to optimizing maximum likelihood

$$\mathbb{E}_{x_0 \sim q(x_0)}[\log p_\theta(x_0)]$$

where

$$p_\theta(x_0) = p_\theta(x_T) \cdot \prod_{t=0}^{T-1} p_\theta(x_t | x_{t+1})$$

Diffusion Models

After a lot of simplifications, the training objective simplifies* down to

$$\mathbb{E}_q \Big[\underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_T|\mathbf{x}_0) \parallel p(\mathbf{x}_T))}_{L_T} + \sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} \underbrace{- \log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$

This is constant

Can ignore this for now

* This objective is actually a lower bound on the maximum likelihood estimation.

Generally referred to as *Evidence Lower Bound (ELBO)*

Diffusion Models

$$\sum_{t>1} \underbrace{D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \parallel p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}}$$

The formulation of the forward process allows a close form expression for the left distribution, and it is a Gaussian distribution.

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$
$$\text{where} \quad \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \quad \text{and} \quad \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

We know that the right distribution can also be parameterized as a Gaussian distribution, hence, this KL divergence is essentially a problem of matching the mean of the two distributions.

Diffusion Models

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

where $\boxed{\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t}$ and $\tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$

We basically need to learn this mapping.

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(x_t; \boxed{\mu_\theta(x_t, t)}, \sigma_\theta(x_t, t))$$

Diffusion Models

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}),$$

where $$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t$$

$$p_\theta(x_t|x_{t+1}) = \mathcal{N}(x_t; \mu_\theta(x_t, t), \sigma_\theta(x_t, t))$$

Recall that x_t and x_0 are related in the following way:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \epsilon\sqrt{(1-\bar{\alpha}_t)}$$

Given x_t and t as input, the only unknown quantity in the target is x_0 (or equivalently the input noise \epsilon). With further simplification, the objective function becomes

$$\mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t)\right\|^2\right]$$

Diffusion Models

$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

Weighted version of the original ELBO objective

$$L_{\text{simple}}(\theta) \coloneqq \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

Diffusion Models

$$\mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\frac{\beta_t^2}{2\sigma_t^2\alpha_t(1-\bar{\alpha}_t)}\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

Weighted version of the original ELBO objective

$$L_{\text{simple}}(\theta) \coloneqq \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon}-\boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0+\sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},t)\right\|^2\right]$$

Recall the final objective function from score matching. They are equivalent!

$$\sum_{i=0}^{N}\mathbb{E}_{\epsilon\sim\mathcal{N}(0,I),\, x\sim p(x)}[||\epsilon - s'_\theta(x+\epsilon\sigma_i,\sigma_i)||_2^2]$$

There is an equivalence between a weighted MLE objective of diffusion models and score matching.

Recent paper have shown that under certain weightings, score matching objective is equivalent to ELBO objective function.

Diffusion Models

Score-Based Generative Modeling through Stochastic Differential Equations

Highly recommend reading this work that formalizes the connection between diffusion models and score based generative models.

Diffusion Models

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Diffusion Models

Algorithm 2 Sampling

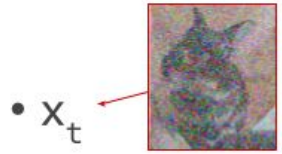
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

During sampling, every step, we have some noisy image x_t, and our model is predicting the noise that was added to x_t (i.e. effectively denoising the image).

From score conditioning, we also know that this predicted “noise” is effectively related to* score of the true data distribution.

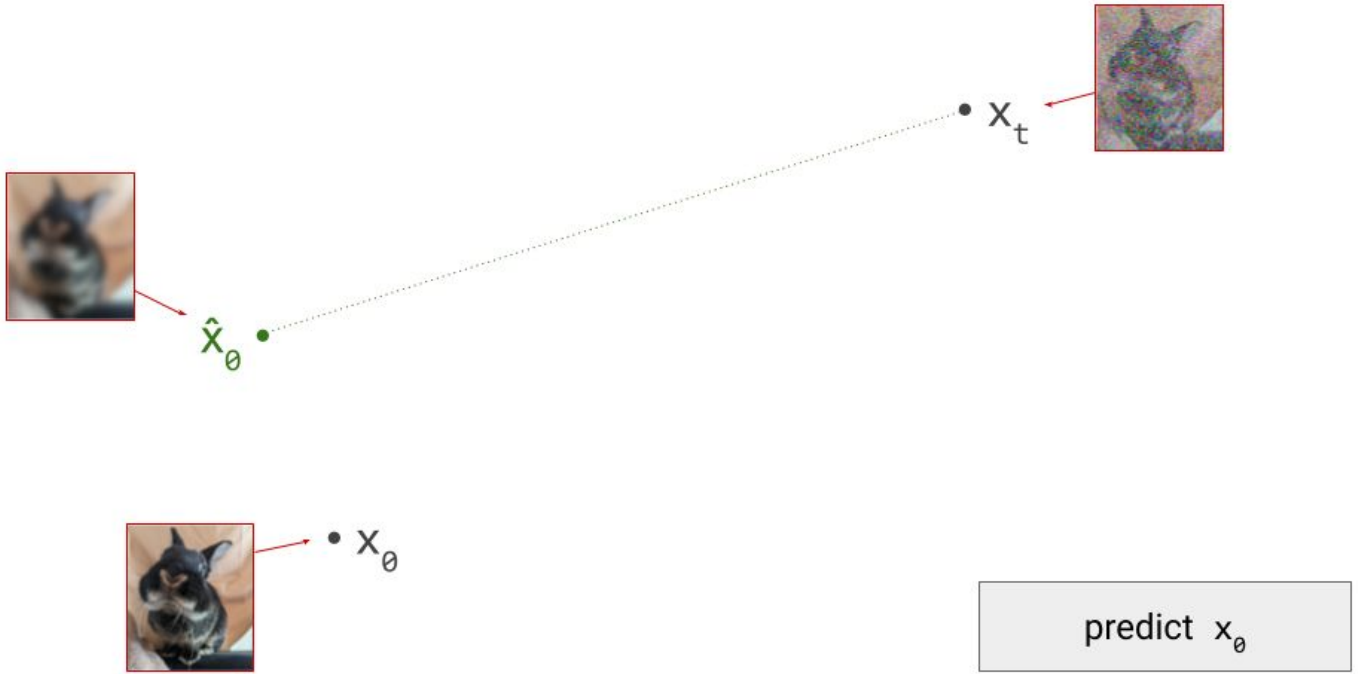
*actually this points in the opposite direction of score

Diffusion Models



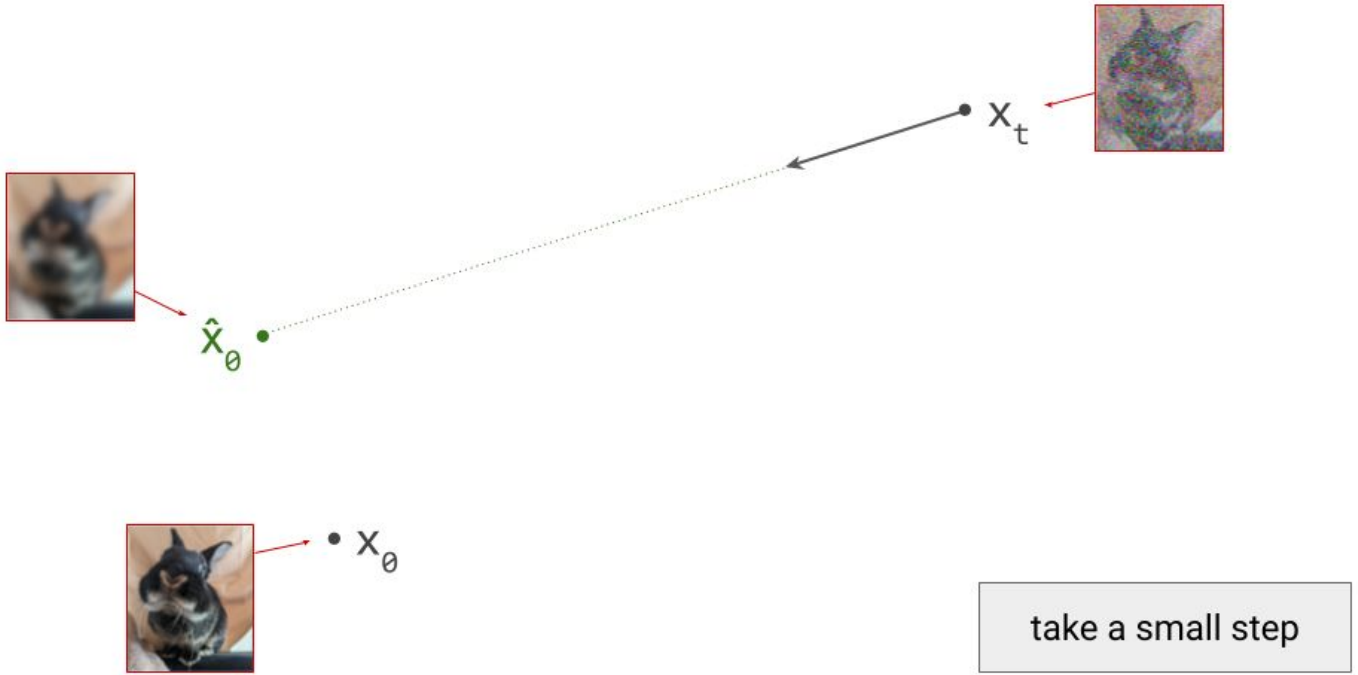
The geometry of diffusion guidance

Diffusion Models



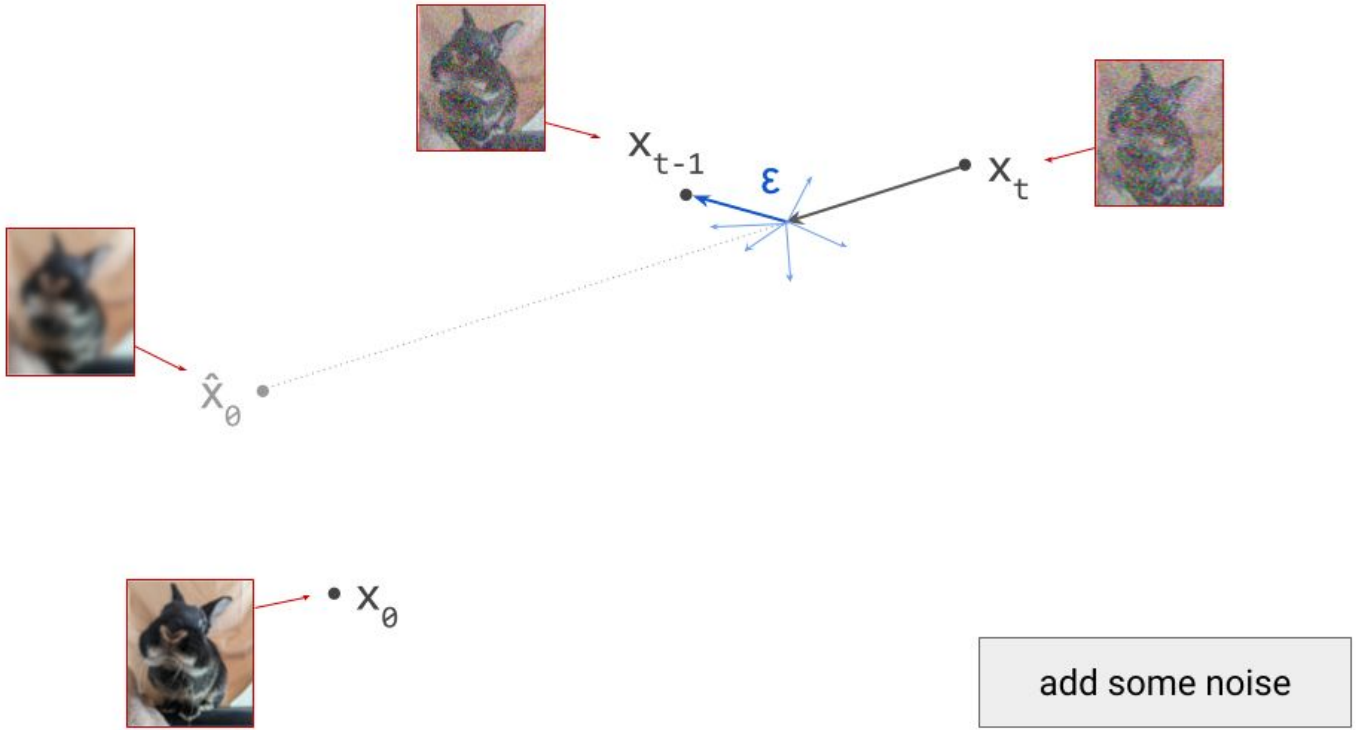
The geometry of diffusion guidance

Diffusion Models



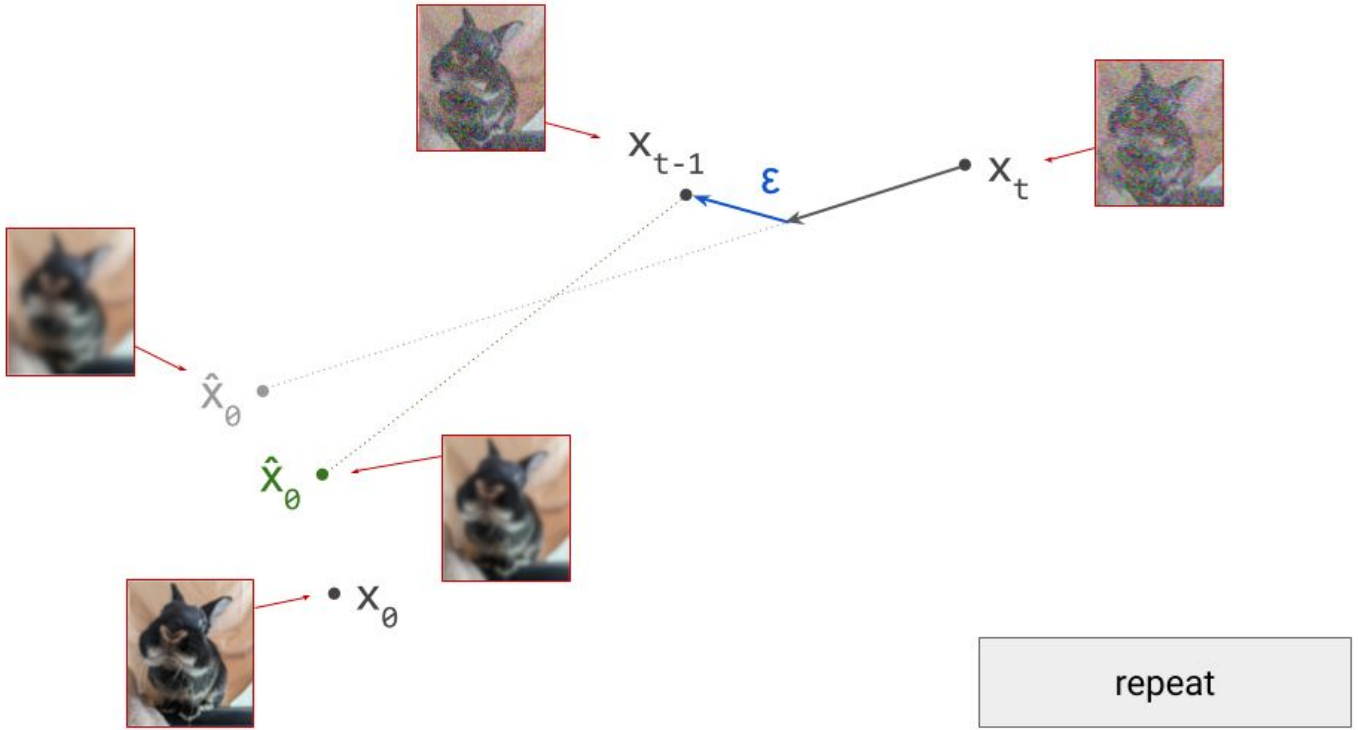
The geometry of diffusion guidance

Diffusion Models



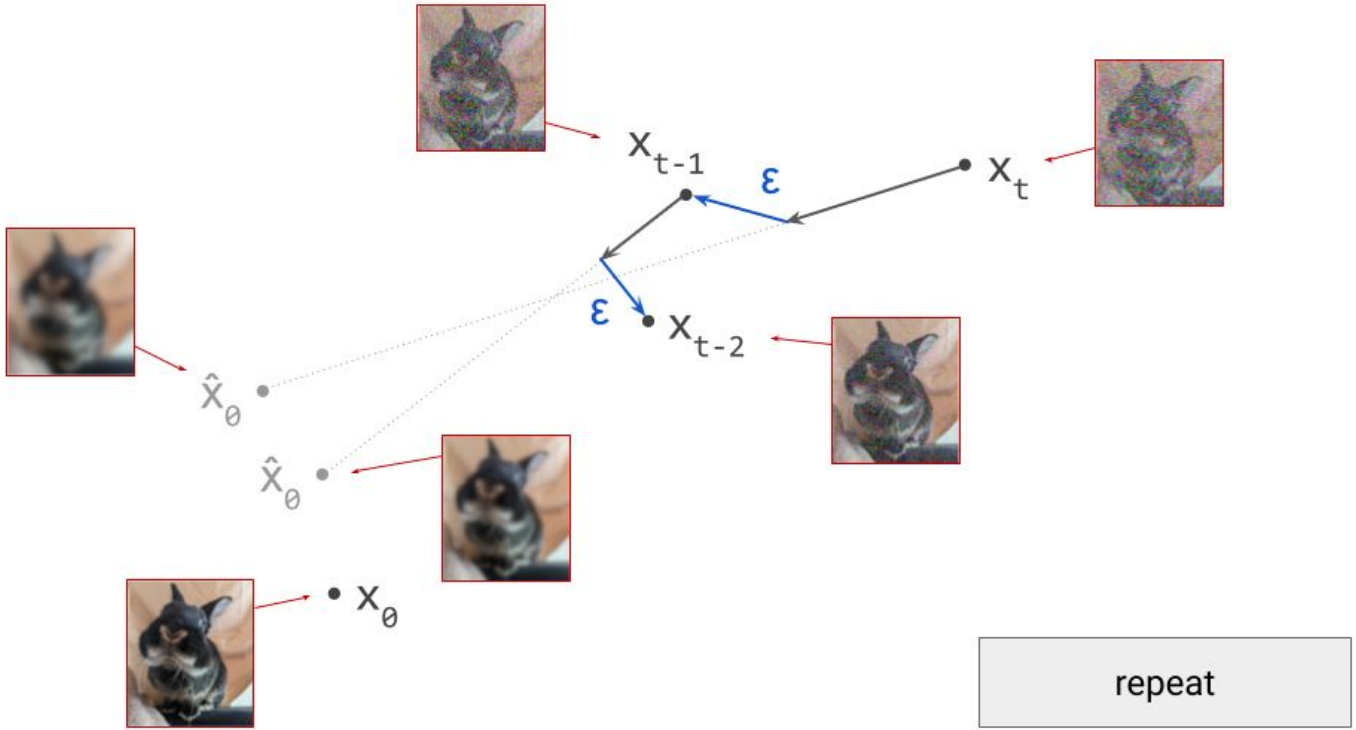
The geometry of diffusion guidance

Diffusion Models



The geometry of diffusion guidance

Diffusion Models



The geometry of diffusion guidance

Diffusion Models

Diffusion Models

$$x_{t-1} = A_t x_t + B_t * \boxed{\epsilon_\theta(x_t, t)} + C_t z$$

$$x_{t-1} = A_t x_t + B'_t * \boxed{\nabla_{x_t} \log p(x_t)} + C_t z$$

Diffusion model / score matching sampling can be thought of as taking steps in the direction of high probability region of data distribution.

Hence, if we run this sampling for several iterations, we should end up with a sample which has high log likelihood under data distribution.

Conditional Diffusion Models

So far, our denoiser network does not get any extra conditioning signal. Generally, in generative models, we want some kind of control during inference.

$$\epsilon_\theta(x_t, t) \rightarrow \epsilon_\theta(x_t, t, c)$$

Examples of such controls / conditioning signals:

- Low resolution image (Super-Resolution model)
- Text (Text to Image model)
- Black and white image (Colorization model)

Conditional Diffusion Models

$$x_{t-1} = A_t x_t + B_t * \boxed{\epsilon_\theta(x_t, t, c)} + C_t z$$

$$x_{t-1} = A_t x_t + B'_t * \boxed{\nabla_{x_t} \log p(x_t|c)} + C_t z$$

Classifier Guidance

$$x_{t-1} = A_t x_t + B'_t * \nabla_{x_t} \log p(x_t|c) + C_t z$$

For a given conditioning signal c, say we have access to a classifier that models the probability *p(c | x_t)*.

During sampling from the diffusion model, we can add this additional term to further enforce the diffusion model to specifically follow the conditioning signal.

$$x_{t-1} = A_t x_t + B'_t * [\nabla_{x_t} \log p(x_t|c) + \lambda \nabla_{x_t} \log p(c|x_t)] + C_t z$$

Additional gradient direction to maximize $p(c|x_t)$. The weight λ can be thought of as augmenting the classifier as $p(c|x_t) \rightarrow \frac{1}{Z} p(c|x_t)^\lambda$. The higher the value of λ, the more we focus on the modes of the classifier.

Classifier Guidance

Say, we are sampling an image conditioned on the text “dog”. And we have a classifier that classifies images between a “dog” and a “cat”.

This additional classifier guidance essentially guides us towards images where p(“cat” | x_t) is close to 0.

More likely

Low Diversity

Less likely

Diffusion Models Beat GANs on Image Synthesis

Classifier Guidance

What does this classifier look like?

- The classifier should be able to give a probability of the class **c** given a **noisy image x_t.**
- e.g. for class conditional ImageNet, this would be a classifier which takes a noisy image and gives logits for all the 1000 classes.
- for text to image, this could be an autoregressive captioner, that captions a noisy image.

We need to train this classifier for every new task, which can be non-trivial.

Classifier (**Free)** Guidance

Can we estimate this quantity some other way?

$$\nabla_{x_t} \log p(c|x_t)$$

Bayes Rule to the rescue!

$$\log p(c|x_t) = \log p(x_t|c) + \log p(c) - \log p(x_t)$$

$$\nabla_{x_t} \log p(c|x_t) = \boxed{\nabla_{x_t} \log p(x_t|c)} - \boxed{\nabla_{x_t} \log p(x_t)}$$

We already have this. This is what we trained our diffusion model for.

When training the diffusion model, we can drop the conditioning signal some x% of the time. And during sampling, estimate this by just dropping the conditioning signal.

Classifier-Free Diffusion Guidance

Classifier (**Free)** Guidance

$$\log p(c|x_t) = \log p(x_t|c) + \log p(c) - \log p(x_t)$$
$$\nabla_{x_t} \log p(c|x_t) = \nabla_{x_t} \log p(x_t|c) - \nabla_{x_t} \log p(x_t)$$

$$x_{t-1} = A_t x_t + B'_t * [\nabla_{x_t} \log p(x_t|c) + \lambda \nabla_{x_t} \log p(c|x_t)] + C_t z$$

$$x_{t-1} = A_t x_t + B'_t * [(\lambda + 1)\nabla_{x_t} \log p(x_t|c) - \lambda \nabla_{x_t} \log p(x_t)] + C_t z$$

New sampling iteration rule. Every sampling step,
we are running two forward passes of the model.

Conditional Diffusion Models

$$x_{t-1} = A_t x_t + B_t * \boxed{\epsilon_\theta(x_t, t, c)} + C_t z$$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \epsilon \sqrt{(1 - \bar{\alpha}_t)}$$

Recall that $\epsilon_\theta(x_t, t, x)$ is approximating ϵ. So from these two equations, at every sampling step we can extract our current prediction of fully denoised image $\hat{x_0}$.

At the start of sampling, this $\hat{x_0}$ will be a blurry image, as the denoiser is essentially denoising from pure noise. Over time, this prediction will get sharper.

Conditional Diffusion Models

$$x_{t-1} = A_t x_t + B_t * \boxed{\epsilon_\theta(x_t, t, c)} + C_t z$$

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \epsilon \sqrt{(1 - \bar{\alpha}_t)}$$

If our original image data x_0 has a pre-defined range of [-1, 1] or [0, 1] (which is usually the case), we ideally do not want this $\hat{x_0}$ to deviate from that range.

The diffusion models generally do not impose any architecture constraint to enable this, hence this problem quite often arises.

Conditional Diffusion Models

(a) No thresholding. (b) Static thresholding. (c) Dynamic thresholding.

Figure A.9: Thresholding techniques on 256×256 samples for "A photo of an astronaut riding a horse." Guidance weights increase from 1 to 5 as we go from top to bottom. No thresholding results in poor images with high guidance weights. Static thresholding is an improvement but still leads to oversaturated samples. Our dynamic thresholding leads to the highest quality images. See Fig. A.10 for more qualitative comparison.

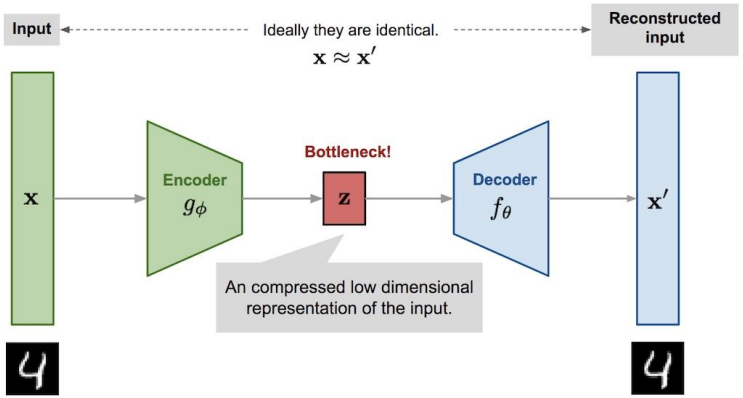
Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding

Diffusion Models in Practice: Continuous Latent Space

Task: High Resolution Image Generation, say generating 1K x 1K images.

Given the resolution is very high to model, we need to compress images into a low dimensional latent space. Unlike autoregressive models, there is no need to discretize representations.

e.g. typically, we can compress by a factor of 16x16, so a 1K x 1K image can be mapped to 128 x 128 latent image.



Diffusion Models in Practice: Cascaded Generation

Task: High Resolution Image Generation, say generating 1K x 1K images.

An alternative approach is cascaded generation, where you train multiple models to progressively generate larger and larger images.

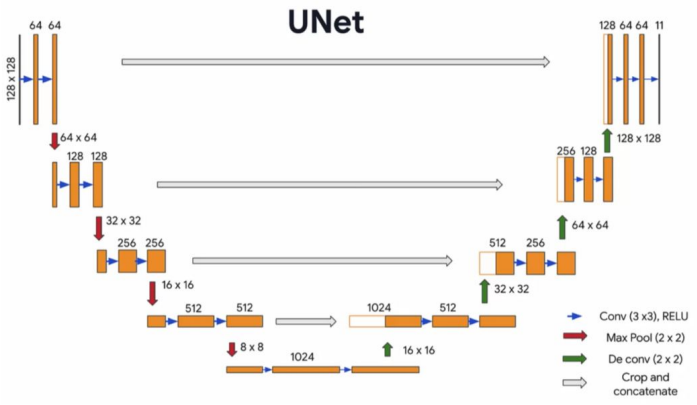
[High Fidelity Image Generation Using Diffusion Models]

Diffusion Models in Practice: Architecture

Only 1 requirement:

The dimensions of input and output should be the same.

Generally two alternatives for architecture:



[U-Net: Convolutional Networks for Biomedical Image Segmentation]

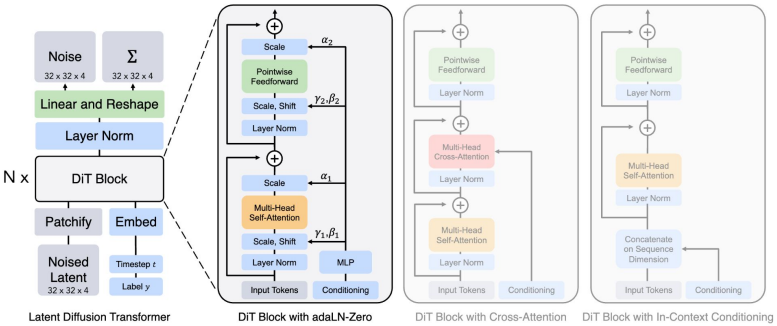


Figure 3. **The Diffusion Transformer (DiT) architecture.** *Left:* We train conditional latent DiT models. The input latent is decomposed into patches and processed by several DiT blocks. *Right:* Details of our DiT blocks. We experiment with variants of standard transformer blocks that incorporate conditioning via adaptive layer norm, cross-attention and extra input tokens. Adaptive layer norm works best.

[Scalable Diffusion Models with Transformers]

Diffusion Models in Practice: Noise Schedule

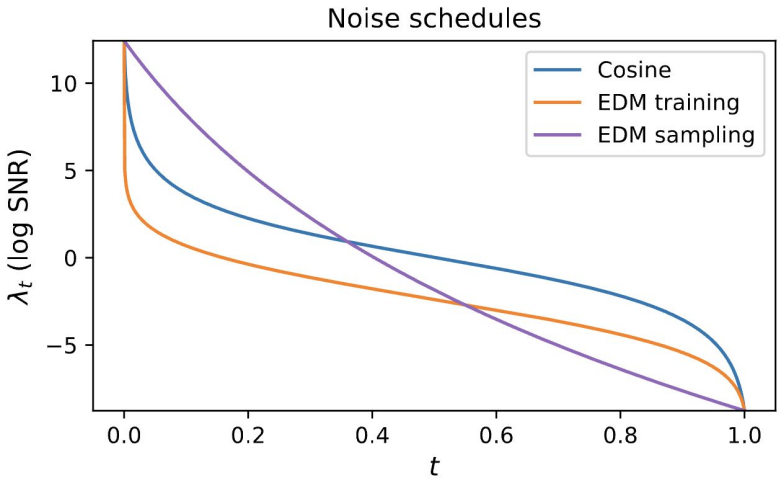
Recall the training objective of diffusion models, where we uniformly sample a timestep t, and based on that, a particular level of noise is added to the image.

Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Typically, we define the function logSNR(t) = $\log\left(\frac{\bar{\alpha}_t}{1-\bar{\alpha}_t}\right)$, which is the signal-to-noise ratio. We can decide any kind of logSNR(t) function for the purpose of training, and this is generally a very important training hyper parameter to tune.

Diffusion Models in Practice: Noise Schedule



Example noise schedules

We can technically use a different noise schedule for training, and a different one for sampling.

[Understanding Diffusion Objectives as the ELBO with Simple Data Augmentation]

Diffusion Models in Practice: Slow Sampling

Typically, diffusion models require 100-1000 sampling iterations to converge to a sample.

This makes them significantly expensive compared to alternatives such GAN, VAE, Normalizing Flows which all require exactly 1 forward pass.

A lot of research has gone into improving diffusion model sampling speed:

Denoising Diffusion Implicit Models → Deterministic Sampler, Smoother trajectory
DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps
DPM-Solver++: Fast Solver for Guided Sampling of Diffusion Probabilistic Models

Diffusion Models in Practice: Slow Sampling

Another direction is model distillation. Effectively, we run diffusion sampling from the teacher model, and train a student model to approximate two step denoising into 1 step denoising.

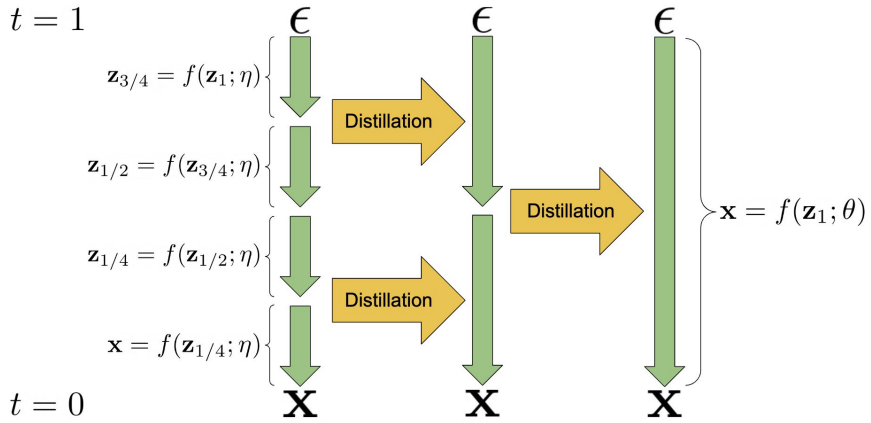


Figure 1: A visualization of two iterations of our proposed *progressive distillation* algorithm. A sampler $f(\mathbf{z}; \eta)$, mapping random noise ϵ to samples $\mathbf{x}$ in 4 deterministic steps, is distilled into a new sampler $f(\mathbf{z}; \theta)$ taking only a single step. The original sampler is derived by approximately integrating the *probability flow ODE* for a learned diffusion model, and distillation can thus be understood as learning to integrate in fewer steps, or *amortizing* this integration into the new sampler.

[Progressive Distillation for Fast Sampling of Diffusion Models]

More References

Elucidating the Design Space of Diffusion-Based Generative Models

Simple diffusion: End-to-end diffusion for high resolution images

Simpler Diffusion (SiD2): 1.5 FID on ImageNet512 with pixel-space diffusion

Analyzing and Improving the Training Dynamics of Diffusion Models

Thank You!